Embedded Linux QA
How to Not Lie With Statistics

Embedded Linux Conference
North America 2018

Prof. Dr. Wolfgang Mauerer
Technical University of Applied Sciences Regensburg
Siemens AG, Corporate Research

About

- Siemens Corporate Technology: Corporate Competence Centre Embedded Linux
- Technical University of Applied Science Regensburg
  - Theoretical Computer Science
  - Head of Digitalisation Laboratory

Target Audience Assumptions

- System Builders & Architects, Software Architects
- Not necessarily RT-Linux and Safety-Critical Linux experts
- No statistical knowledge assumed

Disclaimer

How to not lie with statistics

Disclaimer

~~How to not lie with statistics~~

Disclaimer

~~How to not lie with statistics~~ How to avoid incidentally over-interpreting measured data, presenting data in not so useful ways, and/or drawing unsupported conclusions and making inflated claims from/based on statistical analyses

Outline

Embedded QA: What?

Functional properties

- Speed & Throughput
- Response Latencies
- Test coverage
- …

Non-Functional properties

- Stability
- Availability
- Scaleability
- Correctness ($\approx$ Bugs)
- …

Embedded QA: How? I

Testing

- CI, Load Testing, ...
- Statistics ☞ Process efficiency, gradual change detection

Review

- Manual Inspection of Patches/Pull Requests
- Statistics ☞ Efficiency, Coverage

Embedded QA: How? II

Certification

- SIL/ASIL (Safety), CC (Security), …
- Statistics ☞ Satisfy certification criteria

Formal Verification & Proofs

- Quality “Gold Standard”
- Feasible for small components/portions only
- Statistics ☞ Approximation, WCET estimation, …

Outline

Measuring Properly

- Reproducibility – can others reproduce/interpret results?
- Sufficient Duration – when is certainty achieved?
- Traceability – do we understand what's going on?

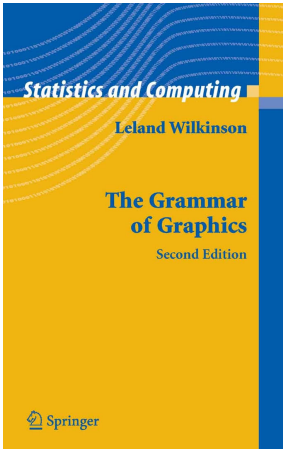
First rule of data analysis

- 80% of effort: cleaning up data
- 20% of effort: analysis

Reproducibility: Tidy Data

1. Each variable forms a column
2. Each observation forms a row
3. Each type of observational unit forms a table (separate file)

GNU R & Grammar of Graphics



- Alternative: Grammar of Graphics in Python: `http://ggplot.yhathq.com/`
- Interactive and reproducible analysis: Jupyter `jupyter.org`

Messy example: Latency measurements ✗

CPU	ID	System	Low Load	High Load	I/O Load	Network Load
ARM	1	Tegra	27	14	22	29
ARM	2	rpi	31	450	50	60
x86	3	qemu	32	50	110	62
…						
ARM	1212346	Tegra	29	41	22	92

Tidy example: Latency measurements ✓

CPU	ID	System	Type	Value
ARM	1	Tegra	Low Load	27
ARM	2	Tegra	High Load	14
ARM	3	Tegra	I/O Load	22
ARM	4	Tegra	Network Load	29
ARM	5	rpi	Low Load	31
…				
ARM	67556356	Tegra	Network Load	92

Tidyverse

- Clean up/transform data
- “Opinionated collection of R packages designed for data science”: `www.tidyverse.org`

Second rule of data analysis

Never underestimate the value of tidy data. Seriously.

Three ways of understanding data

1. Descriptive analysis
2. Exploratory analysis
3. Confirmatory analysis

Order matters

- Simple analyses before formal test
- Proper visual understanding often more important than explicit calculations

Categorical

- Binary (dead/alive)
- Nominal (colours)
- Ordinal (army ranks, kernel maintainer pecking order)

Quantitative

- Discrete (integers)
- Continuous (reals)

Univariate

One variable per “subject”

Multivariate

Multiple variables per “subject”

Univariate: Scatter plots

- Point-to-point coverage
- Structure, Clusters

Univariate: Density

- Data summary
- Density, Histograms

Multivariate: Basic

- Colours
- Facets, Trellis

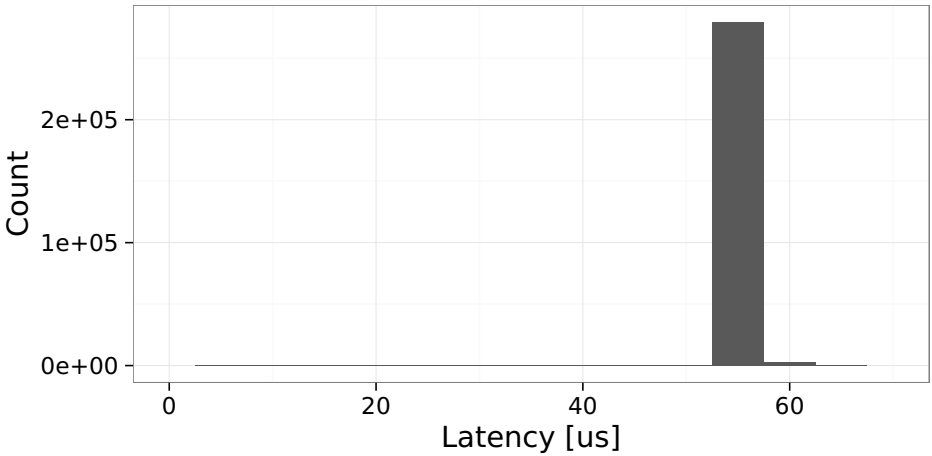
Multivariate: Advanced

- Principal component analysis, self-organising maps, clustering, partitioning
- Visualisation $\Leftrightarrow$ analysis method

Latency Visualisation I

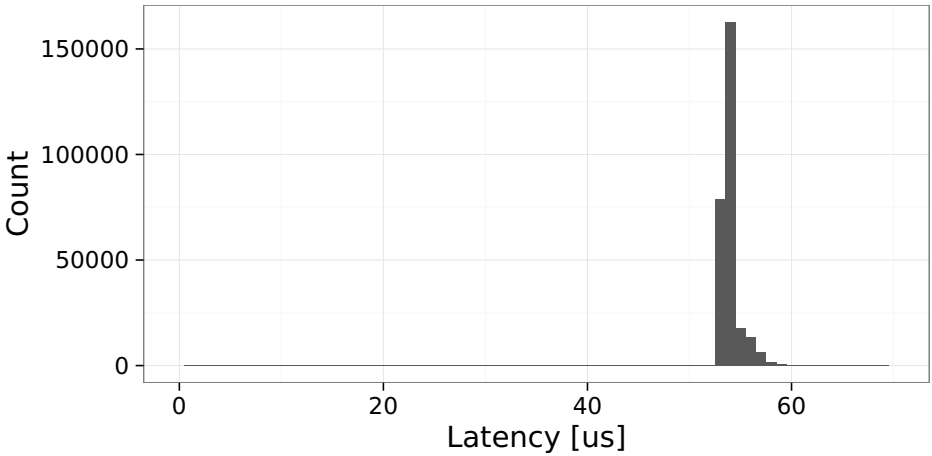
CPU	ID	Latency
0	0	64
0	1	62
0	2	59
1	0	58
1	1	56
2	0	76
2	1	63
…		

Latency Visualisation II



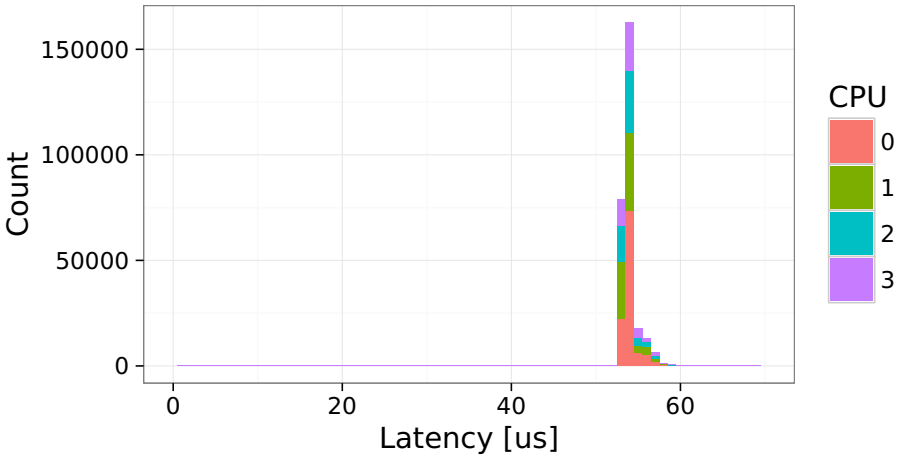

```
ggplot(dat, aes(x=latency)) + geom_histogram()
```

Latency Visualisation II



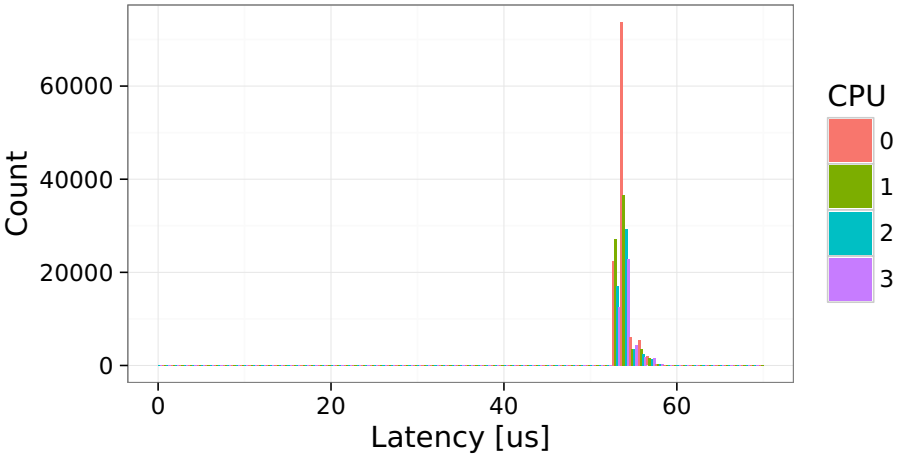

```
ggplot(dat, aes(x=latency)) + geom_histogram(binwidth=1)
```

Latency Visualisation II



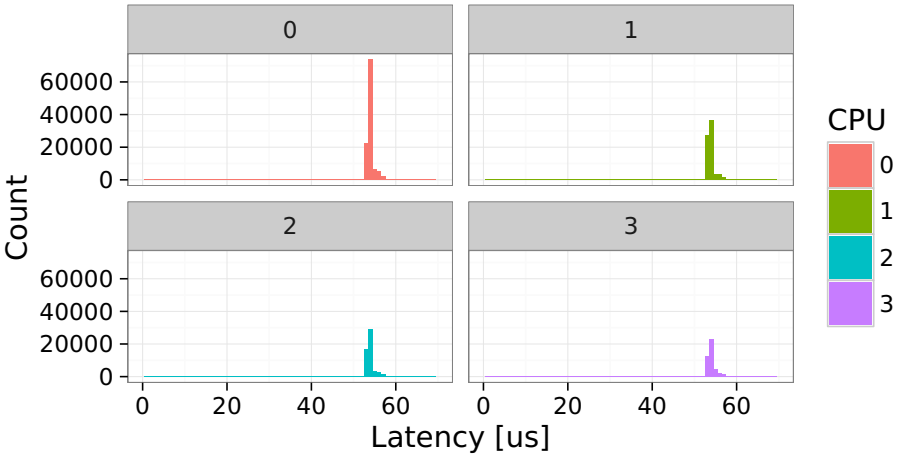

```
ggplot(dat, aes(x=latency, fill=CPU)) + geom_histogram(binwidth=1)
```

Latency Visualisation II



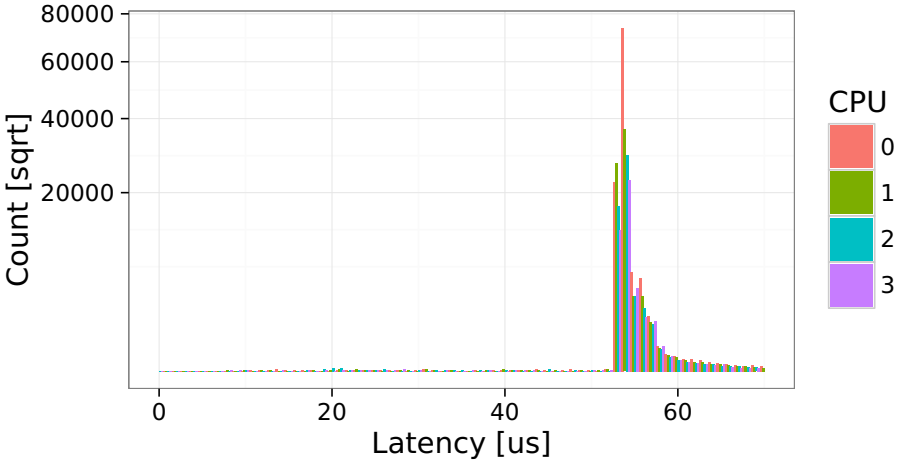

```
ggplot(dat, aes(x=latency, fill=CPU)) +
    geom_histogram(binwidth=1, position="dodge")
```

Latency Visualisation II



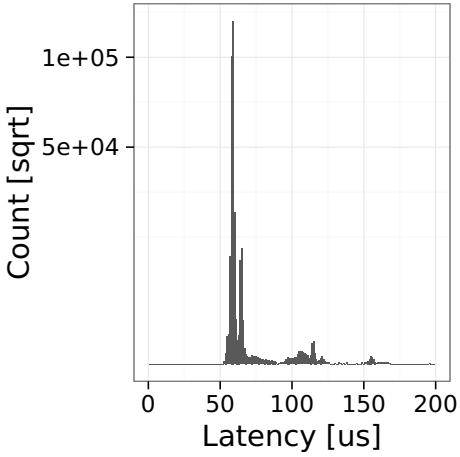

```
ggplot(dat, aes(x=latency, fill=CPU)) +
    geom_histogram(binwidth=1) + facet_wrap(~CPU)
```

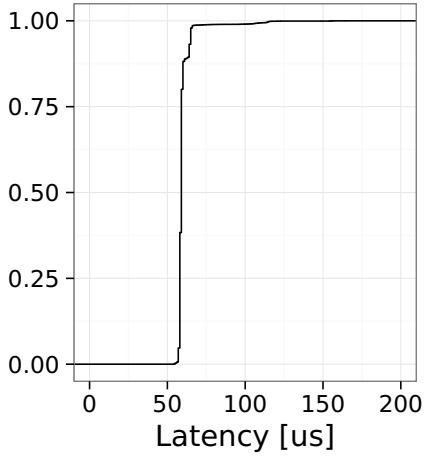
Latency Visualisation II




```
ggplot(dat, aes(x=latency, fill=CPU)) +
    geom_histogram(binwidth=1, position="dodge") + scale_y_sqrt()
```

Latency Visualisation III: Non-Parametric Methods



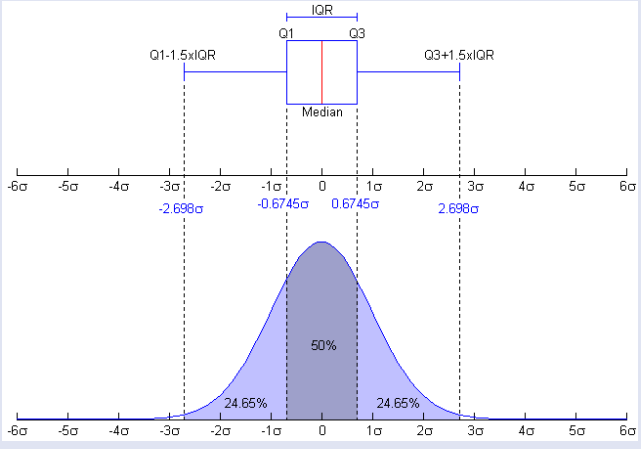



```
ggplot(dat, aes(x=latency)) + stat_ecdf()
```

(Empirical) Cumulative Distribution Function

- Sums up fraction of values that fall into $[0, x]$ at position x
- Parameter free!
- Interpretation requires trained eye

Gaussian Distribution: Mean and SD suffice

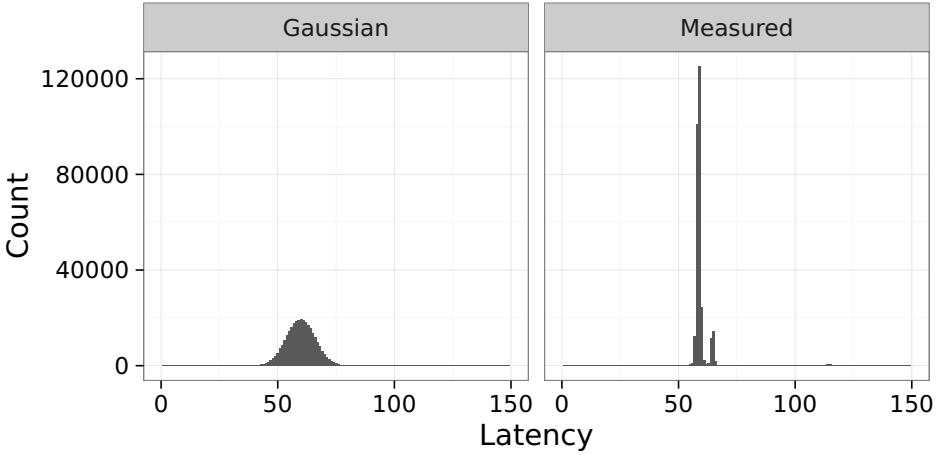


http://en.wikipedia.org/wiki/Probability_density_function

Experiment

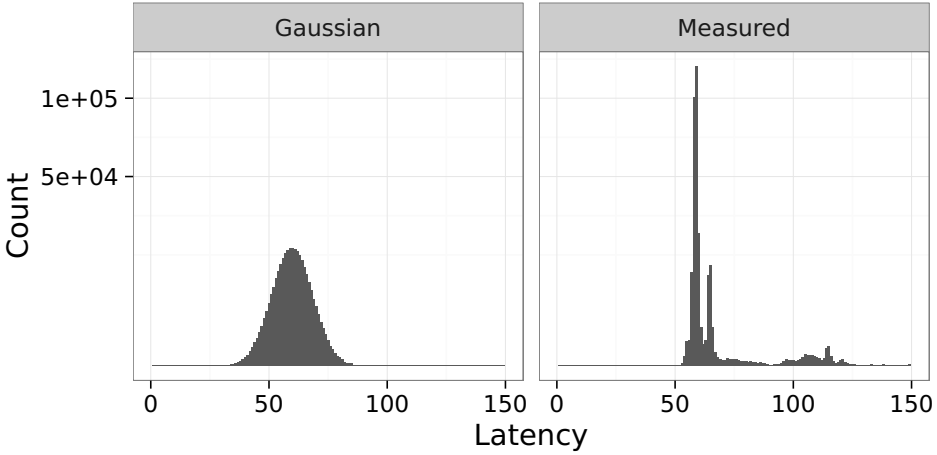
- Gaussian distribution with same mean/SD as measured distribution
- Different stories! (see next slide)

Beware of Summaries III



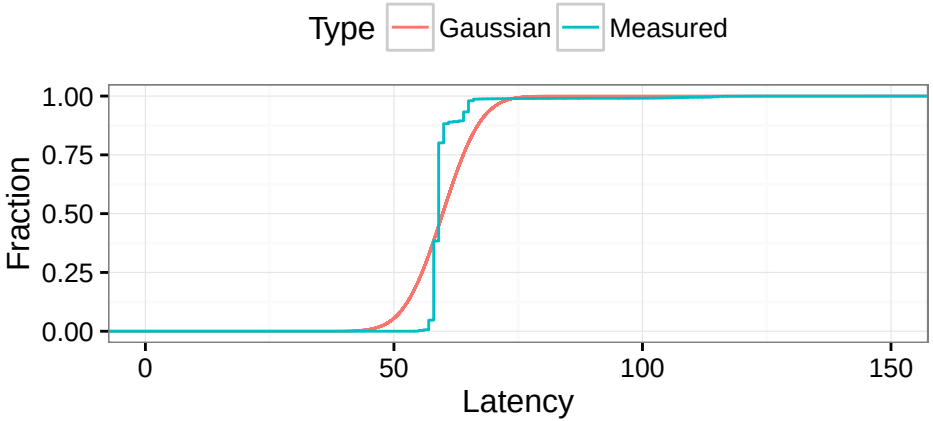

```
ggplot(dat, aes(x=latency)) + geom_histogram(binwidth=1) +
    facet_grid(~type)
```

Beware of Summaries III




```
ggplot(dat, aes(x=latency)) + geom_histogram(binwidth=1) +
    facet_grid(~type) + scale_y_sqrt()
```

Beware of Summaries III




```
ggplot(dat, aes(x=latency, colour=type)) + stat_ecdf()
```

Comparing Datasets/Distributions I

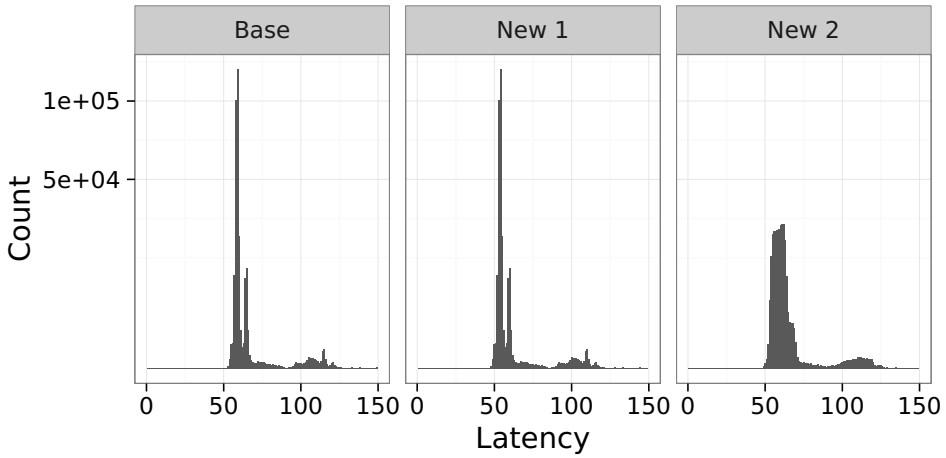
Why?

- Track behavioural changes after system changes
- Load vs. idle behaviour
- Evaluate alternative choices

How?

- Comparing summaries ✗
- Visual inspection/Explorative analysis ✓
- Formal methods/tests ✓✗

System Configuration vs. Latencies



Scenario

1. Base: Typical latency measurement
2. New 1: Averages consistently smaller (5 μs)
3. New 2: Larger variance per sample, but identical average

Point of view

Which one is better?

Formal Tests

- t-test: Check with identical mean values (Gaussian distribution/large sample size)
- Wilcoxon signed-rank test: Test for identical distributions

Visual Tests

- Quantile-Quantile plot ✓✗
- Facetted(!) histograms ✓✗
- Empirical cumulative distribution functions (ecdf) ✓

Conclusion

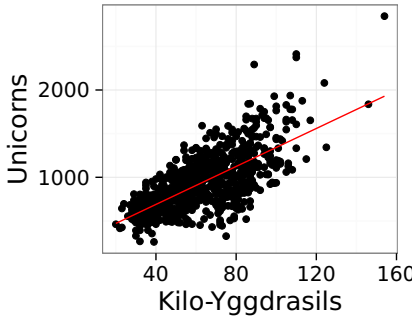
- Formal tests often describe the obvious
- Preconditions not satisfied ☞ formal tests (may be) pointless

Making Predictions: Two Easy Steps

1. Find a (mathematical) model that describes the data
2. Extend the model outside the current (measured) range

Third rule of data analysis

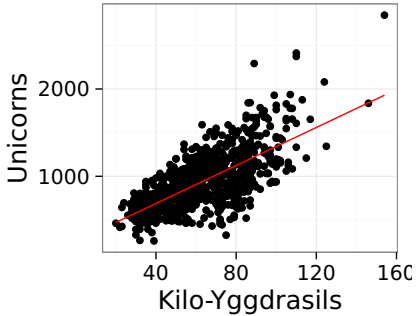
If things sound too good to be true, they probably are not true.



Model

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i, i = 1 \dots N$$

i	# of Observation
β_0	Intercept
β_1	Slope
ε_i	Random deviation



i # of Observation
β_0 Intercept
β_1 Slope
ε_i Random deviation

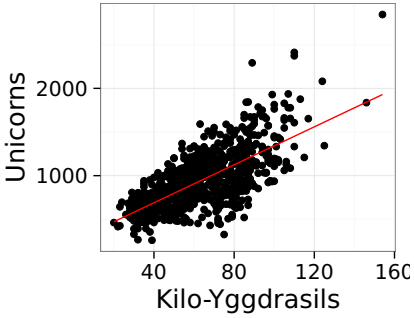
Model

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i, i = 1 \dots N$$

Task

Estimate coefficients $\hat{\beta}_0$, $\hat{\beta}_1$ and $\hat{\epsilon}_i$

Making Predictions II



Model

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i, i = 1 \dots N$$

Solution

Minimise $f(\beta_0, \beta_1) = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2$

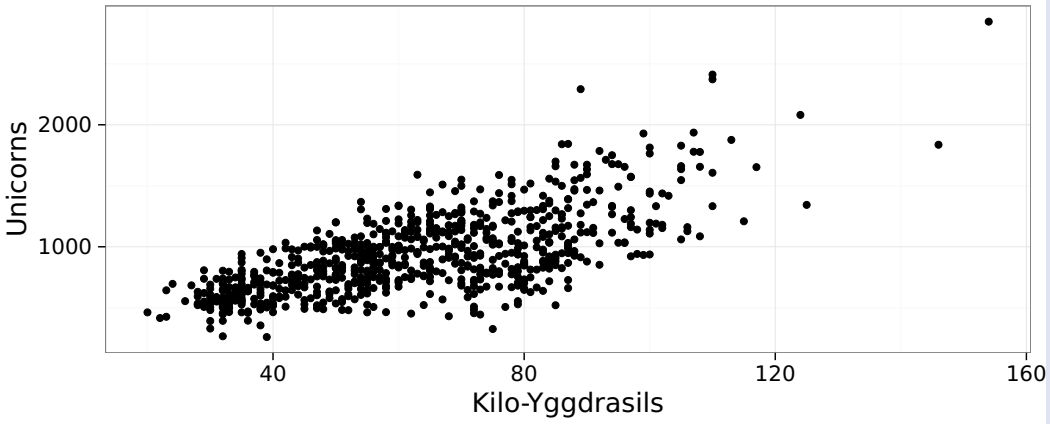
$$\hat{\beta}_0, \hat{\beta}_1 = \arg\min f(\beta_0, \beta_1)$$

i	# of Observation
β_0	Intercept
β_1	Slope
ε_i	Random deviation

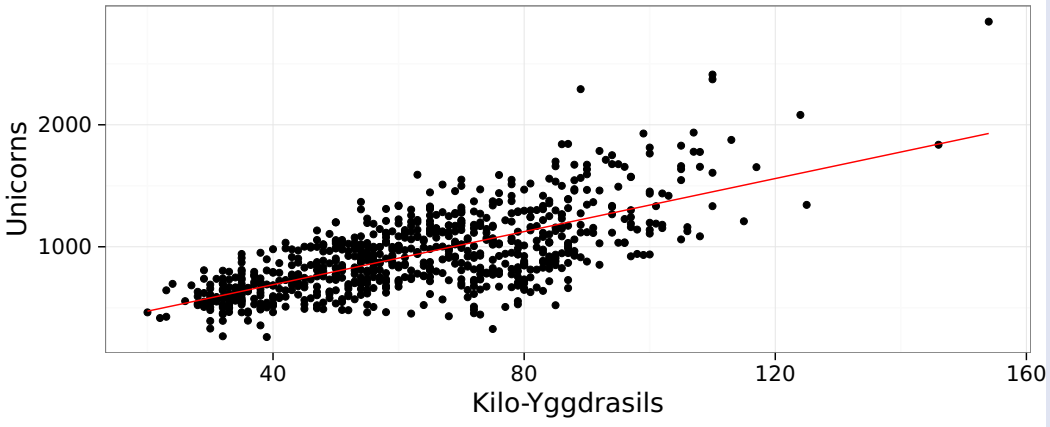
Models and Reality

- No model is correct
- Some models may be useful

Data with Functional Relation

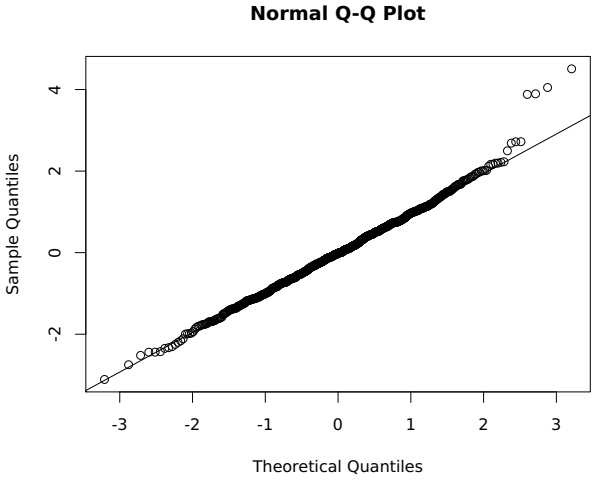


Data with Functional Relation



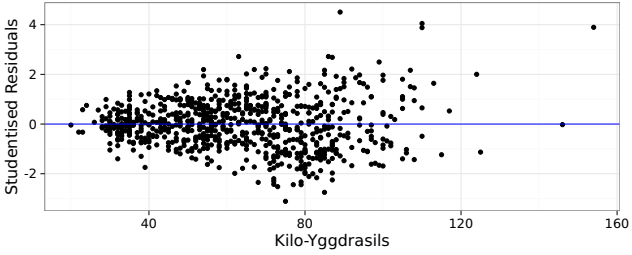
Assumptions (linear regression)

1. Errors are normally distributed: $E(\varepsilon_i) = 0$.



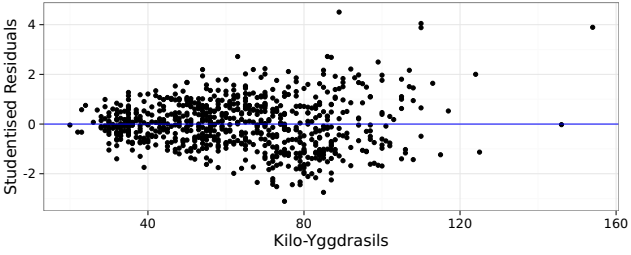
Assumptions (linear regression)

1. Errors are normally distributed: $E(\varepsilon_i) = 0$.
2. Errors are uncorrelated: $\text{cov}(\varepsilon_i, \varepsilon_j) = 0$ for $i \neq j$.



Assumptions (linear regression)

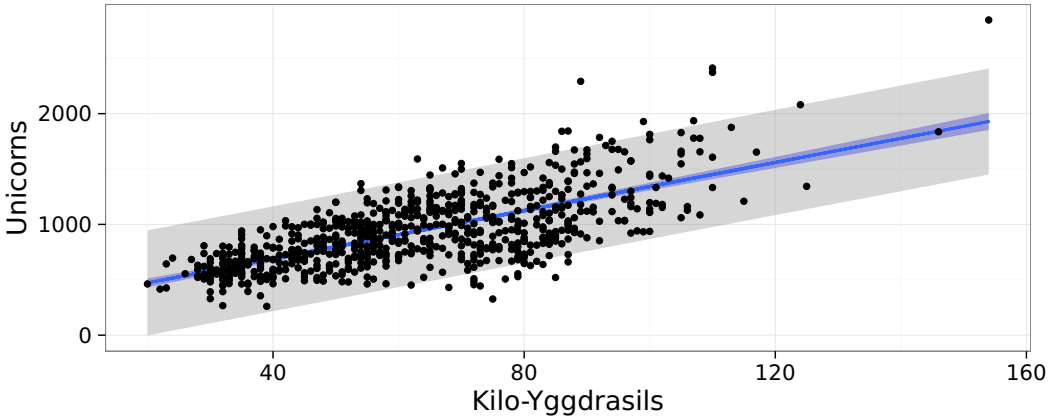
1. Errors are normally distributed: $E(\varepsilon_i) = 0$.
2. Errors are uncorrelated: $\text{cov}(\varepsilon_i, \varepsilon_j) = 0$ for $i \neq j$.
3. Variance of errors is constant: $\text{var}(\varepsilon_i) = \sigma^2$. (homoscedastic vs. heteroscedastic)



Assumptions (linear regression)

1. Errors are normally distributed: $E(\varepsilon_i) = 0$.
2. Errors are uncorrelated: $\text{cov}(\varepsilon_i, \varepsilon_j) = 0$ for $i \neq j$.
3. Variance of errors is constant: $\text{var}(\varepsilon_i) = \sigma^2$. (homoscedastic vs. heteroscedastic)
4. Design matrix (X) has full rank.

Confidence and Prediction Intervals



Use and Abuse of Formal Tests

- Rule of thumb: filter against speculation
- Sample size
- Existence of functional relationship
- Prerequisites of modelling approach
- Statistical significance vs. practical relevance
- Confidence intervals necessary!

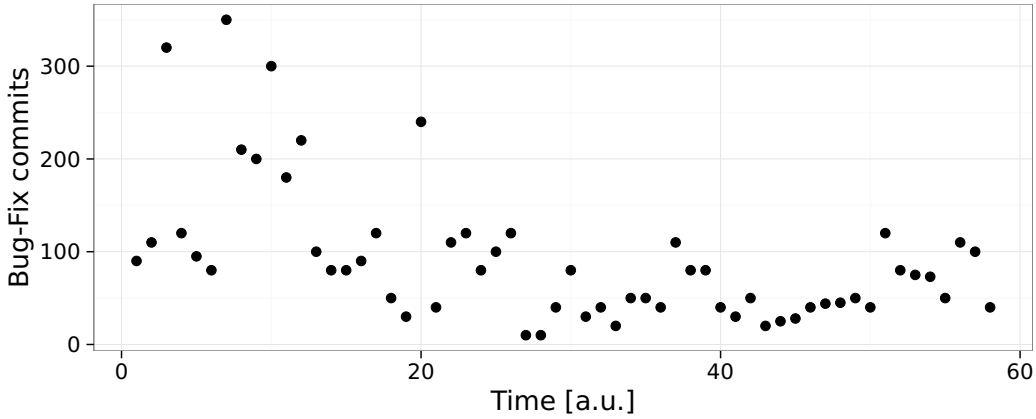
Current Trends

- Quantify quality
  - Software
  - Process
  - Community
- Repository mining

Possible Uses

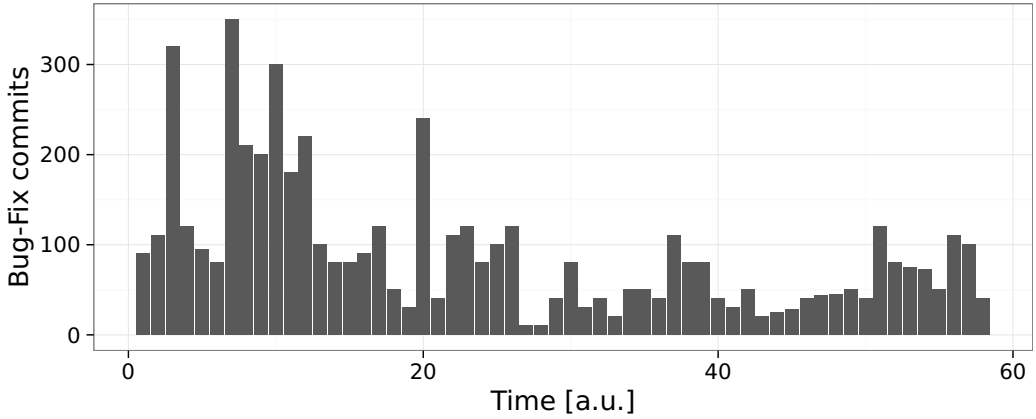
- Predict future behaviour
- Detect weaknesses
- Make quality objective

Example: Fixing Bugs I



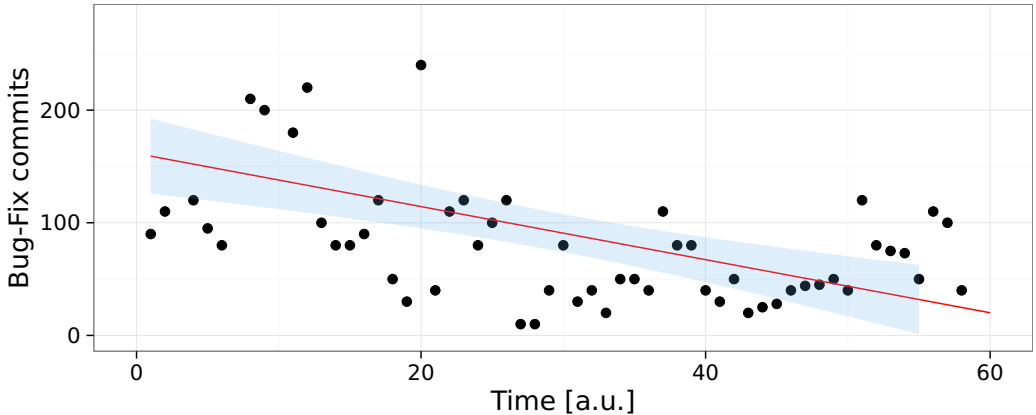

```
ggplot(bug.dat, aes(x=time, y=commits)) + geom_point()
```

Example: Fixing Bugs I

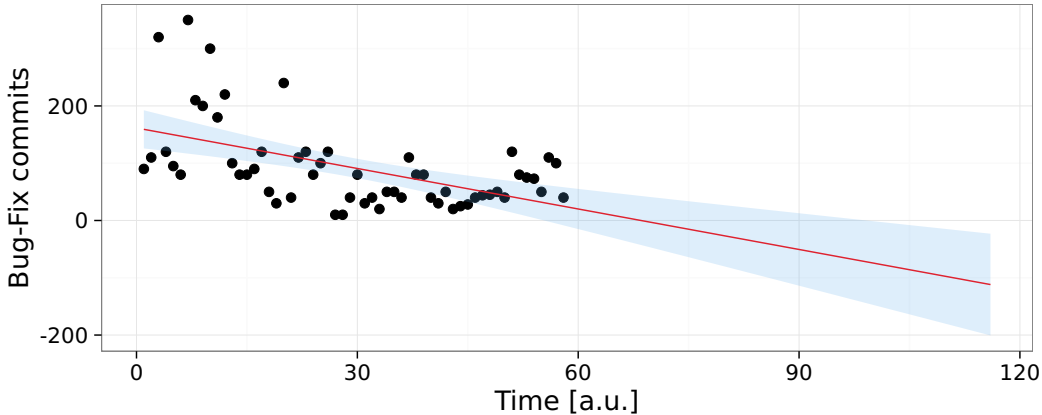



```
ggplot(bug.dat, aes(x=time, y=commits)) +
geom_bar(stat='identity')
```

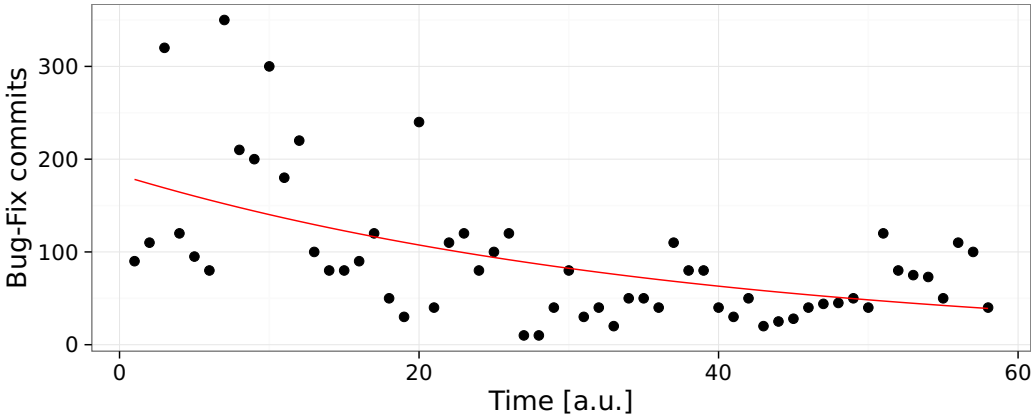
Example: Fixing Bugs II (Linear Model)



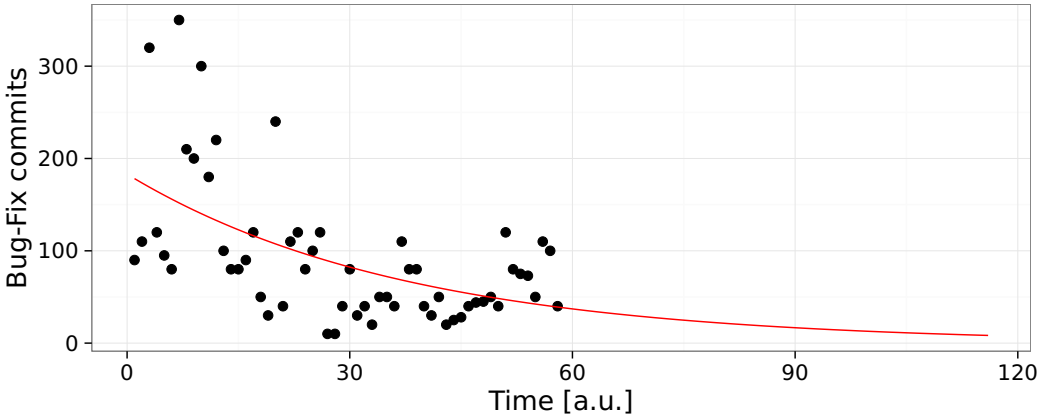
Example: Fixing Bugs II (Linear Model)



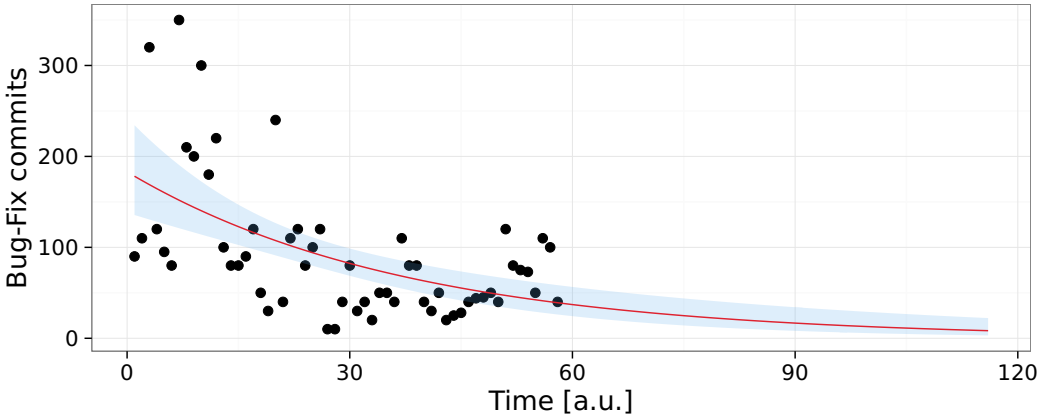
Example: Fixing Bugs III (Generalised Linear Model)



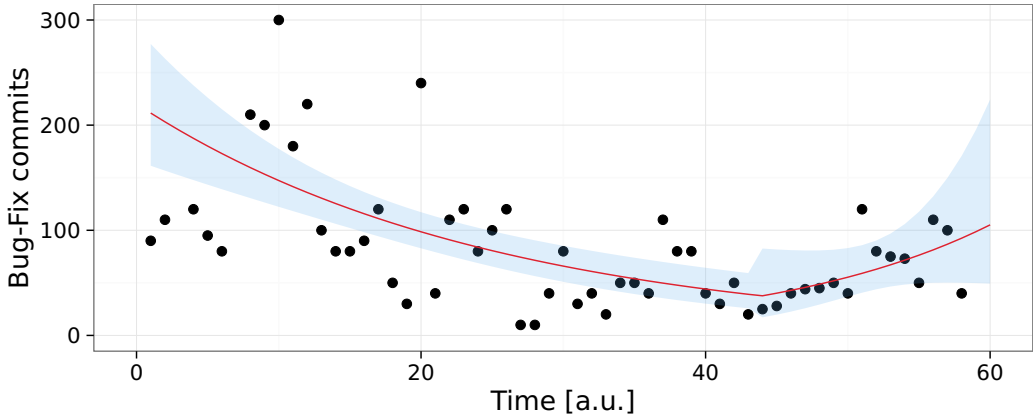
Example: Fixing Bugs III (Generalised Linear Model)



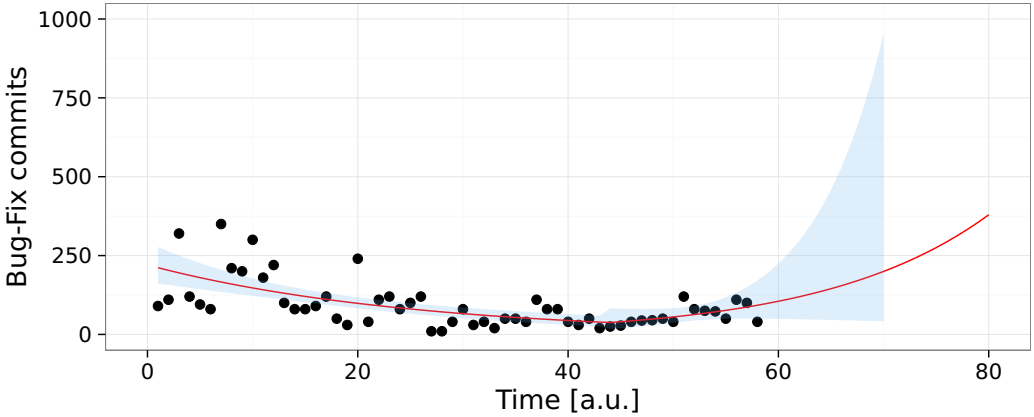
Example: Fixing Bugs III (Generalised Linear Model)



Example: Fixing Bugs IV (Segmented Linear Model)

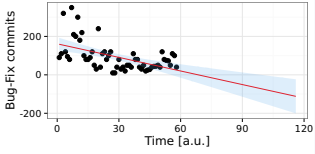


Example: Fixing Bugs IV (Segmented Linear Model)

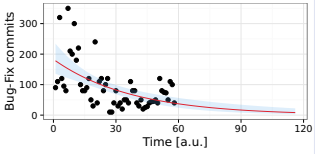


Example: Fixing Bugs V

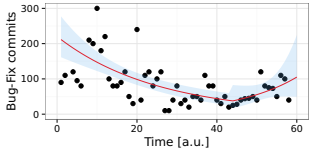
LM



Generalised LM



Segmented LM



Same data, three models

Three (totally) different stories…

Issues

- Base data accuracy?
- Model checking
  - Errors/Residuals: Normally distributed?
  - Variance OK?
- Functional relationship?
- Does the analysis add value beyond
  - common sense?
  - expert interpretation?

Model diagnostics essential

- Find appropriate technique
- Fit and plot model
- Check diagnostic data

Outline

Summary

Living with Data

- “Listen” to the data, understand structures
- Identify important variables & patterns
- Outliers and anomalies, appropriate graph (axis) transformations
- Test underlying assumptions
- Parsimonious models, hypotheses
- Identify mistakes ($\rightarrow$ cleanup)

Don't

- expect miracles
- forget to use expert validation/help!

Summary

Living with Data

- "Listen" to the data, understand structures
- Identify important variables & patterns
- Outliers and anomalies, appropriate graph (axis) transformations
- Test underlying assumptions
- Parsimonious models, hypotheses
- Identify mistakes ($\rightarrow$ cleanup)

Don't

- expect miracles
- forget to use expert validation/help!

☞**Do more statistical analysis! Do less statistical analysis!**

Thanks for your interest!